

[illegible]

EXTI

[illegible]

```

LL               IIIII
LL              IIIIII
                II
LL              II
LL              II
LL              II
LL              II
LL              II
LL              II
LL              II
LL              II
LL              II
LL              II
LLLLLLLLLLLL    IIIIII
LLLLLLLLLLLL    IIIIII
SSSSSSSSS      SS
SSSSSSSSS      SS
                  SS
                  SS
                  SS
                  SS
                    SSSSSS
                    SSSSSS
                              SS
                              SS
                              SS
                              SS
SSSSSSSSS      SSSSSS
SSSSSSSSS      SSSSSS

```

.....


```
1 0001 0 MODULE EXTCONTIG (  
2 0002 0     LANGUAGE (BLISS32),  
3 0003 0     IDENT = 'V04-000'  
4 0004 0 ) =  
5 0005 1 BEGIN  
6 0006 1  
7 0007 1  
8 0008 1 *****  
9 0009 1 *  
10 0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
11 0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
12 0012 1 *  ALL RIGHTS RESERVED.  
13 0013 1 *  
14 0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
15 0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
16 0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
17 0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
18 0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
19 0019 1 *  TRANSFERRED.  
20 0020 1 *  
21 0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
22 0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
23 0023 1 *  CORPORATION.  
24 0024 1 *  
25 0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
26 0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
27 0027 1 *  
28 0028 1 *  
29 0029 1 *****  
30 0030 1  
31 0031 1 ++  
32 0032 1  
33 0033 1 FACILITY: F11ACP Structure Level 2  
34 0034 1  
35 0035 1 ABSTRACT:  
36 0036 1  
37 0037 1     This routine extends a file, keeping it contiguous by actually  
38 0038 1     reallocating and copying the blocks.  
39 0039 1  
40 0040 1 ENVIRONMENT:  
41 0041 1  
42 0042 1     STARLET operating system, including privileged system services  
43 0043 1     and internal exec routines.  
44 0044 1  
45 0045 1 --  
46 0046 1  
47 0047 1  
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 13-Jun-1979 17:39  
49 0049 1  
50 0050 1 MODIFIED BY:  
51 0051 1  
52 0052 1     V03-003 CDS0001 Christian D. Saether 29-Dec-1983  
53 0053 1     Use L_NORM linkage and BIND_COMMON macro.  
54 0054 1  
55 0055 1     V03-002 ACG0367 Andrew C. Goldstein, 26-Oct-1983 19:50  
56 0056 1     Update highwater mark of extended file  
57 0057 1
```



```

: 58      0058 1  | V03-001 STJ3070      Steven T. Jeffreys, 23-Mar-1983
: 59      0059 1  |      Remove unnecessary reference to RETURN_BLOCKS.
: 60      0060 1  |
: 61      0061 1  | V02-004 STJ41739     Steven T. Jeffreys, 24-Nov-1981
: 62      0062 1  |      Explicitly set the allocation control bits when
: 63      0063 1  |      extending the quota file. This will prevent the
: 64      0064 1  |      extend from succeeding when it should have failed.
: 65      0065 1  |
: 66      0066 1  | V02-003 STJ33788     Steven T. Jeffreys, 27-Feb-1981
: 67      0067 1  |      Signal error if extend fails.
: 68      0068 1  |
: 69      0069 1  | B0102  ACG0055       Andrew C. Goldstein, 25-Jul-1979 18:41
: 70      0070 1  |      Interface changes to TRUNCATE_HEADER
: 71      0071 1  |
: 72      0072 1  | B0101  ACG0053       Andrew C. Goldstein, 19-Jul-1979 17:51
: 73      0073 1  |      Disk quota bug fixes
: 74      0074 1  |
: 75      0075 1  |      **
: 76      0076 1  |
: 77      0077 1  |
: 78      0078 1  | LIBRARY 'SYS$LIBRARY:LIB.L32';
: 79      0079 1  | REQUIRE 'SRC$FCPDEF.B32';
: 80      1070 1  |
: 81      1071 1  |
: 82      1072 1  | FORWARD ROUTINE
: 83      1073 1  | EXTEND CONTIG      : L_NORM,      ! extend a file contiguously
: 84      1074 1  | HANDLER;           ! local condition handler
```

```

86      1075 1 GLOBAL ROUTINE EXTEND_CONTIG (FIB, FCB, SIZE) : L_NORM =
87      1076 1
88      1077 1 !++
89      1078 1
90      1079 1 FUNCTIONAL DESCRIPTION:
91      1080 1
92      1081 1 This routine extends a file. If allocated but unused space is
93      1082 1 present, this means simply pushing back the EOF and materializing a
94      1083 1 block of zeroes. If the file is to be physically extended, it is
95      1084 1 copied to a new location on the disk to keep it contiguous.
96      1085 1
97      1086 1 CALLING SEQUENCE:
98      1087 1 EXTEND_CONTIG (ARG1, ARG2, ARG3)
99      1088 1
100     1089 1 INPUT PARAMETERS:
101     1090 1 ARG1: scratch FIB for operation
102     1091 1 ARG2: FCB on which file is open
103     1092 1 ARG3: size by which to extend the file (0 means exponentially)
104     1093 1
105     1094 1 IMPLICIT INPUTS:
106     1095 1 CURRENT_RVN: RVN of current volume
107     1096 1
108     1097 1 OUTPUT PARAMETERS:
109     1098 1 NONE
110     1099 1
111     1100 1 IMPLICIT OUTPUTS:
112     1101 1 PRIMARY_FCB: FCB of file
113     1102 1
114     1103 1 ROUTINE VALUE:
115     1104 1 address of buffer containing next block to use
116     1105 1
117     1106 1 SIDE EFFECTS:
118     1107 1 file extended, storage map altered, FCB & windows altered
119     1108 1
120     1109 1 --
121     1110 1
122     1111 2 BEGIN
123     1112 2
124     1113 2 LINKAGE
125     1114 2 L_MAKE_POINTER = CALL :
126     1115 2 GLOBAL (MAP_POINTER = 9);
127     1116 2
128     1117 2 MAP
129     1118 2 FIB : REF BBLOCK, ! address of FIB for this operation
130     1119 2 FCB : REF BBLOCK; ! address of FCB for file
131     1120 2
132     1121 2 BUILTIN
133     1122 2 ROT,
134     1123 2 FP;
135     1124 2
136     1125 2 GLOBAL REGISTER
137     1126 2 MAP_POINTER = 9 : REF BBLOCK; ! pointer to current retrieval pointer
138     1127 2
139     1128 2 LOCAL
140     1129 2 HEADER : REF BBLOCK, ! address of file file header
141     1130 2 NEXT_VBN, ! next file VBN to use
142     1131 2 NEW_SIZE, ! size to extend file to
```



```
143      1132 2      NEW_LBN,      ! starting LBN of new space
144      1133 2      BUFFER,      ! buffer address of current file block
145      1134 2      NEXT_LBN;      ! LBN of next block to use
146      1135 2
147      1136 2 BIND_COMMON;
148      1137 2
149      1138 2 EXTERNAL ROUTINE
150      1139 2      READ_HEADER      : L_NORM,      ! read file header
151      1140 2      CHARGE_QUOTA    : L_NORM,      ! charge space to user's quota
152      1141 2      ALLOC_BLOCKS    : L_NORM,      ! allocate blocks from storage map
153      1142 2      MAKE_POINTER    : L_MAKE_POINTER, ! build header map pointer
154      1143 2      READ_BLOCK      : L_NORM,      ! read a disk block
155      1144 2      RESET_LBN       : L_NORM,      ! assign new LBN to buffer
156      1145 2      WRITE_BLOCK     : L_NORM,      ! write block to disk
157      1146 2      CREATE_BLOCK    : L_NORM,      ! fabricate a block buffer
158      1147 2      INVALIDATE      : L_NORM,      ! invalidate a buffer
159      1148 2      TRUNCATE_HEADER : L_NORM,      ! truncate file header
160      1149 2      CHECKSUM        : L_NORM,      ! compute file header checksum
161      1150 2      WRITE_HEADER    : L_NORM,      ! write file header
162      1151 2      INIT_FCB2       : L_NORM,      ! update file control block
163      1152 2      ZERO_WINDOWS    : L_NORM;      ! invalidate related file windows
164      1153 2
165      1154 2
166      1155 2 ! Set up context and read the file header. Note that the file must be contiguous.
167      1156 2 !
168      1157 2
169      1158 2 CH$MOVE (FIB$S_FID, FCB[FCB$W_FID], FIB[FIB$W_FID]);
170      1159 2 PRIMARY_FCB = .FCB;
171      1160 2 IF .FCB[FCB$L_STLBN] EQL 0
172      1161 2 THEN ERR_EXIT (SS$ FILESTRUCT);
173      1162 2 HEADER = READ_HEADER (0, .FCB);
174      1163 2
175      1164 2 ! The next VBN to use is the current file eof block number. If the block
176      1165 2 ! is not present in the file, the file must be physically extended.
177      1166 2 !
178      1167 2
179      1168 2 NEW_SIZE = 0;
180      1169 2 NEXT_VBN = .FCB[FCB$L_EFBLK] + 1;
181      1170 2
182      1171 2 IF .NEXT_VBN GTRU .FCB[FCB$L_FILESIZE]
183      1172 2 THEN
184      1173 2 BEGIN
185      1174 2
186      1175 2 ! Compute the number of blocks needed (50% of the current file size),
187      1176 2 ! or as specified if non-zero, and allocate the new space contiguously.
188      1177 2 !
189      1178 2
190      1179 2 IF .SIZE NEQ 0
191      1180 2 THEN NEW_SIZE = .SIZE + .FCB[FCB$L_FILESIZE]
192      1181 2 ELSE NEW_SIZE = .FCB[FCB$L_FILESIZE] + MAXU (.FCB[FCB$L_FILESIZE]/2, 1);
193      1182 2 CHARGE_QUOTA (.HEADER[FH2$ FILEOWNER], .NEW_SIZE - .FCB[FCB$L_FILESIZE],
194      1183 2 BITLIST (QUOTA_CHECK));
195      1184 2
196      1185 2 CLEANUP_FLAGS[CLF_FIXFCB] = 1;
197      1186 2 FIB[FIB$W_EXCTL] = (FIB$M_ALCON OR FIB$M_FILCON);
198      1187 2 IF NOT ALLOC_BLOCKS (.FIB, .NEW_SIZE, NEW_LBN, NEW_SIZE)
199      1188 2 THEN
```

```
200 1189 3 ERR_EXIT (SS$ DEVICEFULL);
201 1190 3 UNREC_COUNT = .NEW_SIZE;
202 1191 3 UNREC_LBN = .NEW_LBN;
203 1192 3 UNREC_RVN = .CURRENT_RVN;
204 1193 3
205 1194 3 ! Now copy the file data from the old file to the newly allocated space.
206 1195 3 !
207 1196 3
208 1197 3 INCR VBN FROM 1 TO .FCB[FCB$L_FILESIZE] DO
209 1198 3 BEGIN
210 1199 3 BUFFER = READ_BLOCK (.VBN + .FCB[FCB$L_STLBN] - 1, 1, DATA_TYPE);
211 1200 3 RESET_LBN (.BUFFER, .VBN + .NEW_LBN - 1);
212 1201 3 WRITE_BLOCK (.BUFFER);
213 1202 3 END;
214 1203 3
215 1204 3 ! Now deallocate the old file blocks. Then build retrieval pointers
216 1205 3 for the new blocks in the file header. Do the truncation with a local
217 1206 3 condition handler enabled for special error recovery.
218 1207 3 !
219 1208 3
220 1209 3 .FP = HANDLER;
221 1210 3 TRUNCATE_HEADER (.FIB, .HEADER);
222 1211 3 .FP = 0;
223 1212 3
224 1213 3 HEADER[FH2$B_MAP_INUSE] = 0;
225 1214 3 CH$FILL (0, 7, HEADER[FH2$B_ACOFFSET] - .HEADER[FH2$B_MPOFFSET])*2,
226 1215 3 .HEADER + .HEADER[FH2$B_MPOFFSET]*2);
227 1216 3 MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2;
228 1217 3 MAKE_POINTER (.NEW_SIZE, .NEW_LBN, .HEADER);
229 1218 3 UNREC_COUNT = 0;
230 1219 3 NEW_SIZE = .NEW_SIZE - .FCB[FCB$L_FILESIZE];
231 1220 3 KERNEL_CALL (ZERO_WINDOWS, .FCB);
232 1221 3 END; ! end of file extension
233 1222 3
234 1223 3 ! Now that we have enough space in the file, push the end of file
235 1224 3 mark back one block and materialize the new block in memory. Also
236 1225 3 update the FCB and flush any windows on it.
237 1226 3 If this file header supports it, stuff the high water field to
238 1227 3 be the allocated size.
239 1228 3 !
240 1229 3
241 1230 3 IF .HEADER [FH2$B_IDOFFSET] GEQU ($BYTEOFFSET (FH2$L_HIGHWATER)+4)/2
242 1231 3 THEN
243 1232 3 HEADER [FH2$L_HIGHWATER] = .NEXT_VBN + 1;
244 1233 3
245 1234 3 BBLOCK [HEADER[FH2$W_RECATTR], FAT$L_EFBLK] = ROT (.NEXT_VBN + 1, 16);
246 1235 3 BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_FFBYTE] = 0;
247 1236 3 KERNEL_CALL (INIT_FCB2, .FCB, .HEADER);
248 1237 3 BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_HIBLK] = .FCB[FCB$L_FILESIZE];
249 1238 3 CHECKSUM (.HEADER);
250 1239 3 WRITE_HEADER ();
251 1240 3 IF .NEW_SIZE NEQ 0
252 1241 3 THEN CHARGE_QUOTA (.HEADER[FH2$L_FILEOWNER], .NEW_SIZE, BITLIST (QUOTA_CHARGE));
253 1242 3
254 1243 3 NEXT_LBN = .FCB[FCB$L_STLBN] + .NEXT_VBN - 1;
255 1244 3 BUFFER = CREATE_BLOCK (.NEXT_LBN, 1, DATA_TYPE);
256 1245 3
```



```
: 257      1246  2 RETURN .BUFFER;  
: 258      1247  2  
: 259      1248  1 END;
```

! end of routine EXTEND_CONTIG

				.TITLE	EXTCONTIG	
				.IDENT	\V04-000\	
				.EXTRN	READ_HEADER, CHARGE_QUOTA	
				.EXTRN	ALLOC_BLOCKS, MAKE_POINTER	
				.EXTRN	READ_BLOCK, RESET_LBN	
				.EXTRN	WRITE_BLOCK, CREATE_BLOCK	
				.EXTRN	INVALIDATE, TRUNCATE_HEADER	
				.EXTRN	CHECKSUM, WRITE_HEADER	
				.EXTRN	INIT_FCB2, ZERO_WINDOWS	
				.PSECT	\$CODE\$,NOWRT,2	
			03FC 00000	.ENTRY	EXTEND_CONTIG, Save R2,R3,R4,R5,R6,R7,R8,R9	1075
		5E	08 C2 00002	SUBL2	#8, SP	
04	A0	24	04 AC 7D 00005	MOVQ	FIB, R0	1158
		08	06 28 00009	MOVC3	#6, 36(R1), 4(R0)	
		AA	08 AC D0 0000F	MOVL	FCB, 8(BASE)	1159
		50	08 AC D0 00014	MOVL	FCB, R0	1160
			30 A0 D5 00018	TSTL	48(R0)	
			05 12 0001B	BNEQ	1\$	
			08C0 8F BF 0001D	CHMU	#2240	1161
			04 00021	RET		
			08 AC DD 00022 1\$:	PUSHL	FCB	1162
			7E D4 00025	CLRL	-(SP)	
		0000G	02 FB 00027	CALLS	#2, READ_HEADER	
		CF	50 D0 0002C	MOVL	R0, HEADER	
		57	6E D4 0002F	CLRL	NEW_SIZE	1168
		50	08 AC D0 00031	MOVL	FCB, R0	1169
56	3C	A0	01 C1 00035	ADDL3	#1, 60(R0), NEXT_VBN	
	38	A0	56 D1 0003A	CMPL	NEXT_VBN, 56(R0)	1171
			03 1A 0003E	BGTRU	2\$	
			00EE 31 00040	BRW	9\$	
			0C AC D5 00043 2\$:	TSTL	SIZE	1179
			08 13 00046	BEQL	3\$	
6E	0C	AC	38 A0 C1 00048	ADDL3	56(R0), SIZE, NEW_SIZE	1180
			0F 11 0004E	BRB	5\$	
51	38	A0	02 C7 00050 3\$:	DIVL3	#2, 56(R0), R1	1181
			03 12 00055	BNEQ	4\$	
		51	01 D0 00057	MOVL	#1, R1	
		6E	38 B041 9E 0005A 4\$:	MOVAB	@56(R0)[R1], NEW_SIZE	
			01 DD 0005F 5\$:	PUSHL	#1	1183
7E	04	AE	38 A0 C3 00061	SUBL3	56(R0), NEW_SIZE, -(SP)	1182
			3C A7 DD 00067	PUSHL	60(HEADER)	
		0000G	03 FB 0006A	CALLS	#3, CHARGE_QUOTA	
		6A	02 88 0006F	BISB2	#2, (BASE)	1185
		50	04 AC D0 00072	MOVL	FIB, R0	1186
	16	A0	05 B0 00076	MOVW	#5, 22(R0)	
			SE DD 0007A	PUSHL	SP	1187
			08 AE 9F 0007C	PUSHAB	NEW_LBN	
			08 AE DD 0007F	PUSHL	NEW_SIZE	
			04 AC DD 00082	PUSHL	FIB	

0000G	CF	04	FB	00085	CALLS	#4, ALLOC_BLOCKS	1189
	05	50	E8	0008A	BLBS	R0, 6\$	
		0850	8F	0008D	CHMU	#2128	
			04	00091	RET		
28	AA		6E	00092	6\$: MOVL	NEW_SIZE, 40(BASE)	1190
24	AA	04	AE	00096	MOVL	NEW_LBN, 36(BASE)	1191
2C	AA	A0	AA	0009B	MOVL	-96(BASE), 44(BASE)	1192
	50	08	AC	000A0	MOVL	FCB, R0	1197
	53	38	A0	000A4	MOVL	56(R0), R3	
			52	000A8	CLRL	VBN	
			2E	000AA	BRB	8\$	
			04	000AC	7\$: PUSHL	#4	1199
			01	000AE	PUSHL	#1	
50	50	08	AC	000B0	MOVL	FCB, R0	
	52	30	A0	000B4	ADDL3	48(R0), VBN, R0	
		FF	A0	000B9	PUSHAB	-1(R0)	
0000G	CF		03	000BC	CALLS	#3, READ_BLOCK	
	58		50	000C1	MOVL	R0, BUFFER	
50	52	04	AE	000C4	ADDL3	NEW_LBN, VBN, R0	1200
		FF	A0	000C9	PUSHAB	-1(R0)	
			58	000CC	PUSHL	BUFFER	
0000G	CF		02	000CE	CALLS	#2, RESET_LBN	1201
			58	000D3	PUSHL	BUFFER	
0000G	CF		01	000D5	CALLS	#1, WRITE_BLOCK	
	52		53	000DA	8\$: AOBLER	R3, VBN, 7\$	1197
CE	6D	0000V	CF	000DE	MOVAB	HANDLER, (FP)	1209
			57	000E3	PUSHL	HEADER	1210
		04	AC	000E5	PUSHL	FIB	
0000G	CF		02	000E8	CALLS	#2, TRUNCATE_HEADER	
			6D	000ED	CLRL	(FP)	1211
		3A	A7	000EF	CLRB	58(HEADER)	1213
	51	01	A7	000F2	MOVZBL	1(HEADER), R1	1214
	50	02	A7	000F6	MOVZBL	2(HEADER), R0	
	50		51	000FA	SUBL2	R1, R0	
	50		02	000FD	MULL2	#2, R0	
	59	6741	3E	00100	MOVAB	(HEADER)[R1], R9	1215
50	6E	00	2C	00104	MOVCS	#0, (SP), #0, R0, (R9)	
			69	00109			
	50	01	A7	0010A	MOVZBL	1(HEADER), R0	1216
	59	6740	3E	0010E	MOVAB	(HEADER)[R0], MAP_POINTER	1217
			57	00112	PUSHL	HEADER	
		08	AE	00114	PUSHL	NEW_LBN	
		08	AE	00117	PUSHL	NEW_SIZE	
0000G	CF		03	0011A	CALLS	#3, MAKE_POINTER	
		28	AA	0011F	CLRL	40(BASE)	1218
	50	08	AC	00122	MOVL	FCB, R0	1219
	6E	38	A0	00126	SUBL2	56(R0), NEW_SIZE	
			50	0012A	PUSHL	R0	1220
0000G	CF		01	0012C	CALLS	#1, ZERO_WINDOWS	
	28		67	00131	9\$: CMPB	(HEADER), #40	1230
			05	00134	BLSSU	10\$	
4C	A7	01	A6	00136	MOVAB	1(R6), 76(HEADER)	1232
	50	01	A6	0013B	MOVAB	1(R6), R0	1234
1C	A7		10	0013F	ROTL	#16, R0, 28(HEADER)	
		20	A7	00144	CLRW	32(HEADER)	1235
			57	00147	PUSHL	HEADER	1236
		08	AC	00149	PUSHL	FCB	

EXTE
V04-

Address	Op Code	Op Name	Comments	Hex
0000G	CF	02 FB 0014C	CALLS #2, INIT_FCB2	1237
50	50	AC D0 00151	FCB, R0	1238
1A	A7	A0 B0 00155	56(R0), 26(HEADER)	1239
0000G	CF	57 DD 0015A	PUSHL HEADER	1240
0000G	CF	01 FB 0015C	CALLS #1, CHECKSUM	1241
		00 FB 00161	CALLS #0, WRITE_HEADER	
		6E D5 00166	TSTL NEW_SIZE	
		0D 13 00168	BEQL 11\$	
		02 DD 0016A	PUSHL #2	
04	AE	DD 0016C	PUSHL NEW_SIZE	
3C	A7	DD 0016F	PUSHL 60(HEADER)	
0000G	CF	03 FB 00172	CALLS #3, CHARGE_QUOTA	
50	50	AC D0 00177	FCB, R0	1243
56	56	A0 C1 0017B	48(R0), NEXT_VBN, R0	1244
		04 DD 00180	PUSHL #4	
		01 DD 00182	PUSHL #1	
		70 9F 00184	-(NEXT_LBN)	
0000G	CF	03 FB 00186	CALLS #3, CREATE_BLOCK	
58	58	D0 0018B	R0, BUFFER	
		04 0018E	RET	1248

; Routine Size: 399 bytes, Routine Base: \$CODE\$ + 0000


```
1249 1 ROUTINE HANDLER (SIGNAL, MECHANISM) =
1250 1
1251 1 ++
1252 1
1253 1 FUNCTIONAL DESCRIPTION:
1254 1
1255 1 This routine is the condition handler for file extension. It is
1256 1 enabled only during the truncate call (deallocating the old file
1257 1 blocks). Normal error handling would cause the entire file to
1258 1 be dropped on the floor. Since we already have a new good copy, we
1259 1 should forge ahead. Note that no error status is returned to the user,
1260 1 although we will log a system error.
1261 1
1262 1
1263 1 CALLING SEQUENCE:
1264 1 HANDLER (ARG1, ARG2)
1265 1
1266 1 INPUT PARAMETERS:
1267 1 ARG1: address of signal array
1268 1 ARG2: address of mechanism array
1269 1
1270 1 IMPLICIT INPUTS:
1271 1 FILE_HEADER: address of file file header
1272 1
1273 1 OUTPUT PARAMETERS:
1274 1 NONE
1275 1
1276 1 IMPLICIT OUTPUTS:
1277 1 NONE
1278 1
1279 1 ROUTINE VALUE:
1280 1 $$$_RESIGNAL or none if unwind
1281 1
1282 1 SIDE EFFECTS:
1283 1 file header map area cleaned out
1284 1
1285 1 --
1286 1
1287 2 BEGIN
1288 2
1289 2 MAP
1290 2 SIGNAL : REF BBLOCK, ! signal array arg
1291 2 MECHANISM : REF BBLOCK; ! mechanism array arg
1292 2
1293 2
1294 2 ! Check the condition code for FCP error exit and check that it is not a
1295 2 ! write error. Then initialize the header's map area and unwind. On other
1296 2 ! signals we simply resignal.
1297 2 !
1298 2
1299 2 IF .SIGNAL[CHFSL SIG_NAME] EQL $$$ CMODUSER
1300 2 THEN $UNWIND (DEPADR = MECHANISM[CHFSL_MCH_DEPTH]);
1301 2
1302 2 RETURN $$$_RESIGNAL; ! status is irrelevant if unwind
1303 2
1304 1 END; ! end of routine handler
```

						.EXTRN		SYSSUNWIND		
				0000	00000	HANDLER:	.WORD	Save nothing		1249
	50	04	AC	D0	00002		MOVL	SIGNAL, R0		1299
00000424	8F	04	A0	D1	00006		CMPL	4(R0), #1060		
			0E	12	0000E		BNEQ	1\$		
			7E	D4	00010		CLRL	-(SP)		1300
7E	08	AC	08	C1	00012		ADDL3	#8, MECHANISM, -(SP)		
00000000G	00		02	FB	00017		CALLS	#2, SYSSUNWIND		
	50	0918	8F	3C	0001E	1\$:	MOVZWL	#2328, R0		1302
				04	00023		RET			1304

```
; Routine Size: 36 bytes,    Routine Base: $CODE$ + 018F
```

```

; 317      1305  1
; 318      1306  1 END
; 319      1307  0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	435	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]LIB.L32:1	18619	45	0	1000	00:02.0

COMMAND QUALIFIERS

```
BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:EXTCONTIG/OBJ=OBJ$:EXTCONTIG MSRC$:EXTCONTIG/UPDATE=(ENH$:EXTCONTIG)
```

```
: Size: 435 code + 0 data bytes
: Run Time: 00:21.4
: Elapsed Time: 00:50.1
: Lines/CPU Min: 3669
: Lexemes/CPU-Min: 43544
```


EXTCONTIG
V04-000

F⁴
16-Sep-1984 00:24:08

VAX-11 BLISS-32 V4.0-742

Page 11

; Memory Used: 266 pages
; Compilation Complete

EXT
V04-

0170

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY